24th

THE Machine

**Introduction:**

Industrial environmental conditions have been upgrading day by day with newly developing automation technology. And, as a result of getting rid of the conventional procedures of manufacturing, this leads to an increase in huge workloads. The next-gen industries will be more advanced and automated as compared with existing ones. This brings a new terminology; “Smart Industries”. In this new era, Monitoring as well as controlling of various Industrial applications is challenging as ever. The Internet of Things (IoT), as an emerging technology that brought about rapid advancements in modern technologies, has attracted a lot of attention and is expected to bring benefits to numerous applications. The newly introduced concept is providing a helping hand to achieve Industrial automation through remote access.

So, there is a need for a proper machine monitoring system, as the rate of the production would directly depend on, of course, machines. All monitoring systems have sensors that detect and transmit signals, usually a current or voltage, to the signal processor. The signal processor receives the signal and converts it to a usable output signal. Signal processing may; filter unwanted portions of the input signal, convert signals to digital sets of values, calculate the average, maximum or minimum values of a series of inputs, etc.. The design of signal processors are numerous and varied in purpose. Ultimately, an end-user needs the signal processor output to provide a useful output that can be displayed or used for analysis.

The aim of the project is to monitor any manufacturing plant remotely for temperature, motion, humidity, and carbon dioxide, etc. The project uses an Arduino Mega and ESP8266 and various sensors. With the use of IoT, this project will send the data remotely via an email or an SMS about the current weather and concentration of gases in the plant. The program will set the parameters and if the results are not within the parameters, it will send a warning so that immediate action can be taken accordingly. The prototype may use Blynk as, server and app, to view the data.

Once the project is set up, the motion, gas, and vibration of the machine in the industry can be monitored. Although the monitoring is done automatically when the sensors detect the change in the environment, you can also control the project using the app or through the web address. The data is uploaded to the cloud, and this data can be seen from the web address or the mobile app developed.

**Existing problems Survey**

Some of the factors that may affect the machine’s wear and tear are:

TEMPERATURE :

Temperature affects the processes of deterioration of material gradually and in a variety of ways. Changes in temperature induce a thermal gradient between the surface layer and the inner layer of materials (particularly in materials with lower thermal conductivity), which may result in the degradation of the mechanical properties of the material and can lead to the formation of fine cracks. The formation of cracks is accompanied by a loss of strength and by an increase in material porosity, which may lower the chemical resistance of the material

WIND:

An increase in wind velocity may affect the deterioration of materials in various ways. Wind drives liquid and solid particles from the air to the material surface, where they cause local attrition and contribute to the weathering of the material. The kinetic energy of particles and the degree of inertial impaction of droplets on the material surface =depend on the wind velocity. The high-speed end of the wind spectrum is of interest for abrasion and the low-speed end for diffusion

SALT :

Salt weathering is probably the most important deterioration process of inorganic building materials (such as stone) in cities, but it may be important in coastal areas as well. There are several sources of soluble salts: they can be present in primary materials, or form during weathering processes, or may also penetrate from the outside, e.g. from groundwater or sea spray in coastal areas.

WATER:

Water damage is considered to be one of the essentially problematic elements in the decay of building materials. Water can deteriorate building structures with which it is in continuous contact and can influence the damage to the material surface by acid deposition reactions. To a certain extent, water may also directly affect the quality of building materials such as concrete during its production.

**THE SOLUTION**

**Detection:**

Usually, the machinery installed is quite prone to many external parameters. So we usually have an expiry date which is the estimated time until which the machine shall do its intended function, to its full potential.

But the machine hardly maintains that period due to the unpredictable changes. So our system shall provide an optimal solution that would detect the exact time a machine would stop functioning i.e. real-time monitoring of the machines.

So the detection part includes installation of certain sensors which would detect the following

1) Friction loss

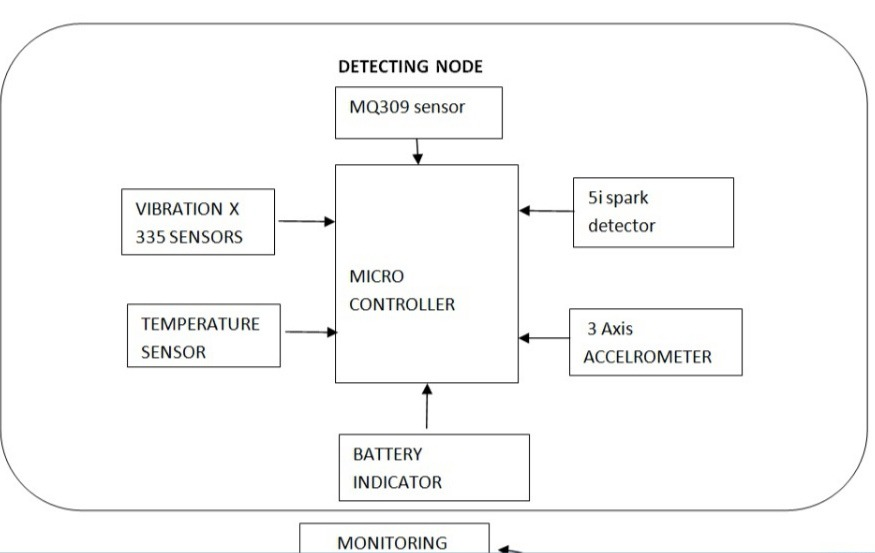
2) Temperature dissipated

3) Power dissipated

4) Rust formation

And any other if required.

The data obtained shall be cumulative and is verified. All sensors shall perform on a threshold level and a real-time day to day analysis is done so that, detailed information regarding the machine is obtained.



For monitoring systems to be complete, assessment criteria must determine when machine owners should be concerned or if immediate action is required. Without assessment criteria, monitoring systems are not needed because their outputs are meaningless. If the assessment criteria are too low, then time and money are wasted. However, if assessment criteria are too high, then human health, environment, and equipment are placed in jeopardy.

**THE END RESULT:**

\* Increased machine availability and reliability

\* Improved operating efficiency

\* Improved risk management (less downtime)

\* Reduced maintenance costs (better planning)

\* Reduced spare parts inventories

\* Improved safety

\* Improved knowledge of the machine condition (safe short-term overloading of machine possible)

\* Extended operational life of the machine

\* Elimination of chronic failures (root cause analysis and redesign)

\* Reduction of post overhaul failures due to improperly performed maintenance or reassembly.

**SENSING AND DETECTION FLOW**

The sensing part includes the following parameters.

1)The vibration of the machine is calculated using the 3-axis accelerometer

2) uneven heat dissipation

3) moisture that is being substituted on the machine.

4) detection of corrosion that’d weaken the machine stability.

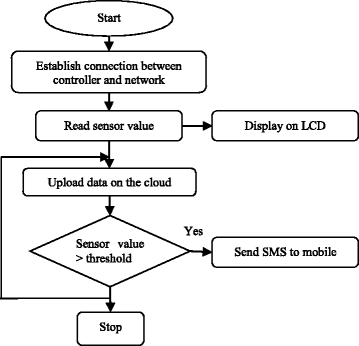
5)MQ309 gas sensor which would detect any gas leakage

6) battery indicator is also prevailed to alert battery usage.

7) 5ispark is used to detect any cuts in insulation if present any.

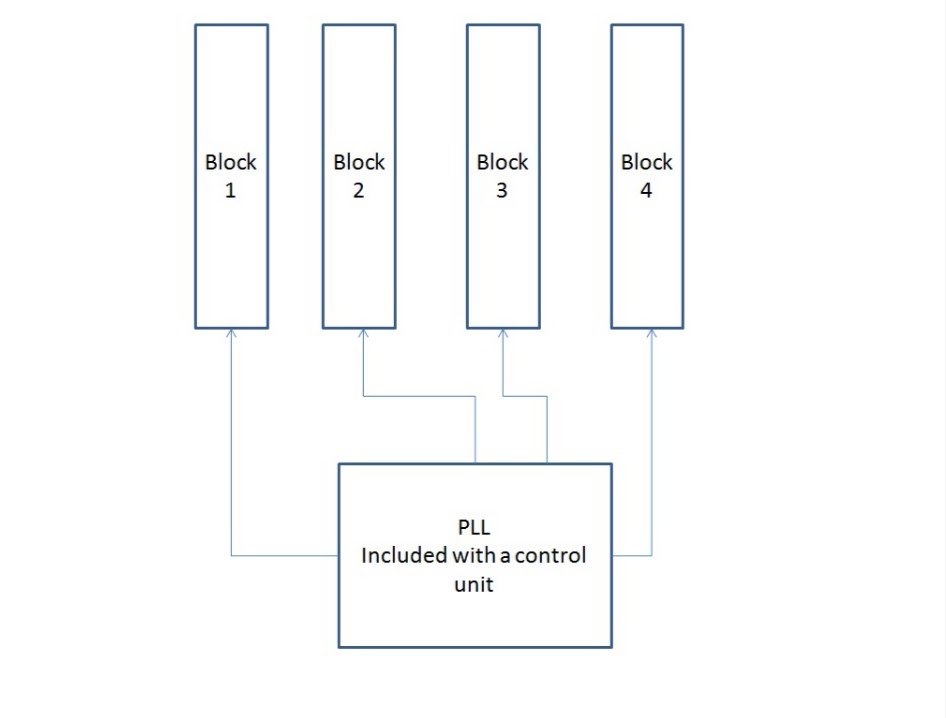
This can be processed by using high-end microcontrollers like the raspberry pi.

FLOWCHART



**Continuous Monitoring( RESULT)** :

Constant or very frequent data collection and analysis are referred to as continuous monitoring. Permanently installed monitoring systems are typically used, with samples and analysis of data done automatically. This type of monitoring is carried out on critical equipment (expensive to replace, with downtime and lost production also being expensive). Changes in condition trigger a more detailed investigation or possibly an automatic shutdown of the equipment.



This particular representation would replicate a processing unit in which the blocks are said to be the mechanism. now from the analysis of how long has been functioning would be calculated then, the PLL or CONTROL UNIT is made up with a certain algorithm which would work in combinations of (1,2) , (2,3),(3,4) and would reset back to the initial stage of (1,2) this algorithm would help to minimize the machine damage which would help in maintaining the machinery for a very long time.

Further involvements are also included such as the concept of semi-automatic interfacing in which the equipment shall be moved into the low power consumption (sleep mode) in which the point is to continue the production at an additional period where human presence is lacking at . this particular process shall be monitored by a certain designed CONTROL SYSTEM UNIT resulting the higher production rate.

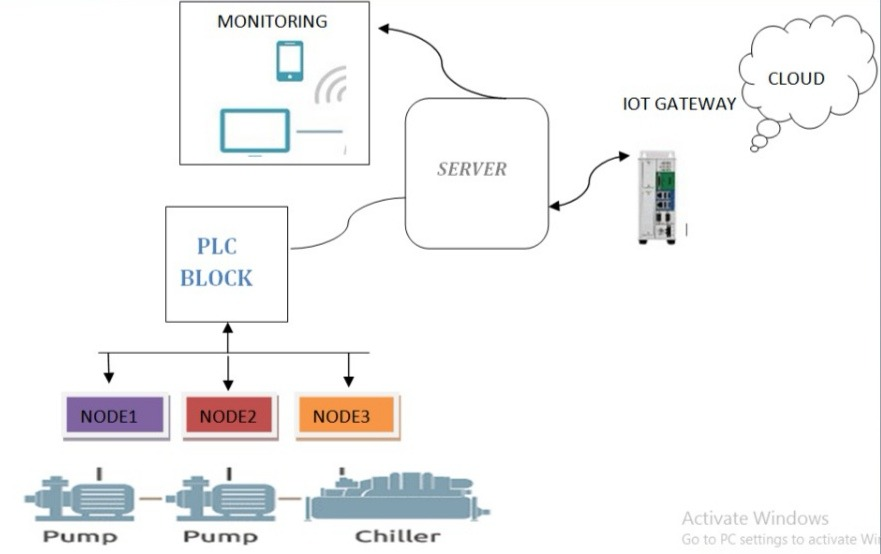
**Conclusion:**

The device contains attachments of sensors which make a complete device now this certain device acts as a nodal point and are placed on the machine surface, the data is sent through this and is given to a gateway (server) where the data is then sent to the cloud. this method ensures total coverage of the machine analysis.

So the after having every single parameter detected the following data is stored and is given to the user with a better visualization in form of graphical etc..

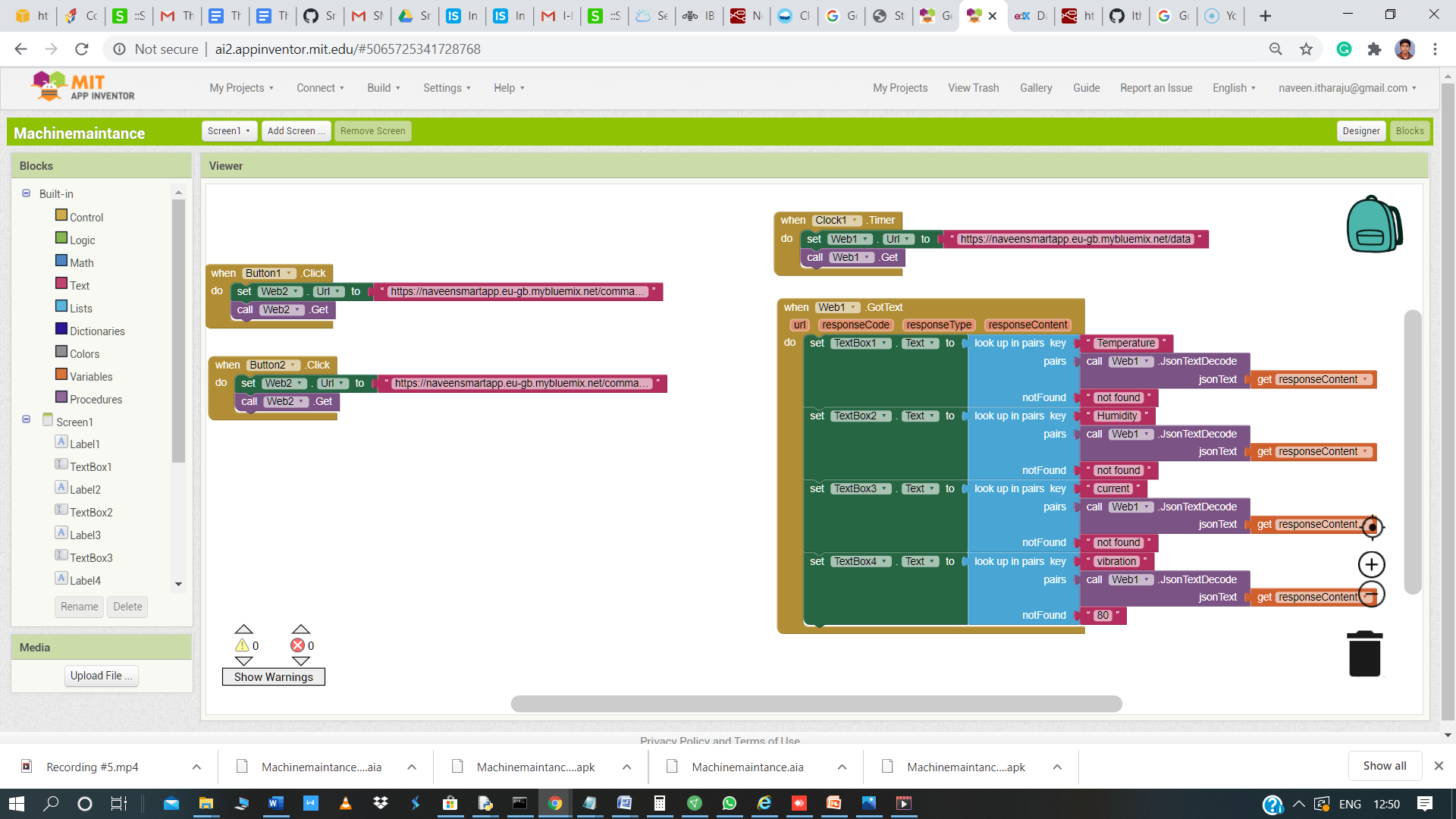
The following data is analyzed and the exact lifetime of the machinery is calculated and along with the production rate of the machine, the exact part of the machine which has gotten faulted is alerted in advance.

So in addition to this, the worker’s safety is also analyzed, if a worker is working in an environment where gases are being emitted from then a timer would be set in such a way that it would alert the worker to finish his shit and pass their shift to others.

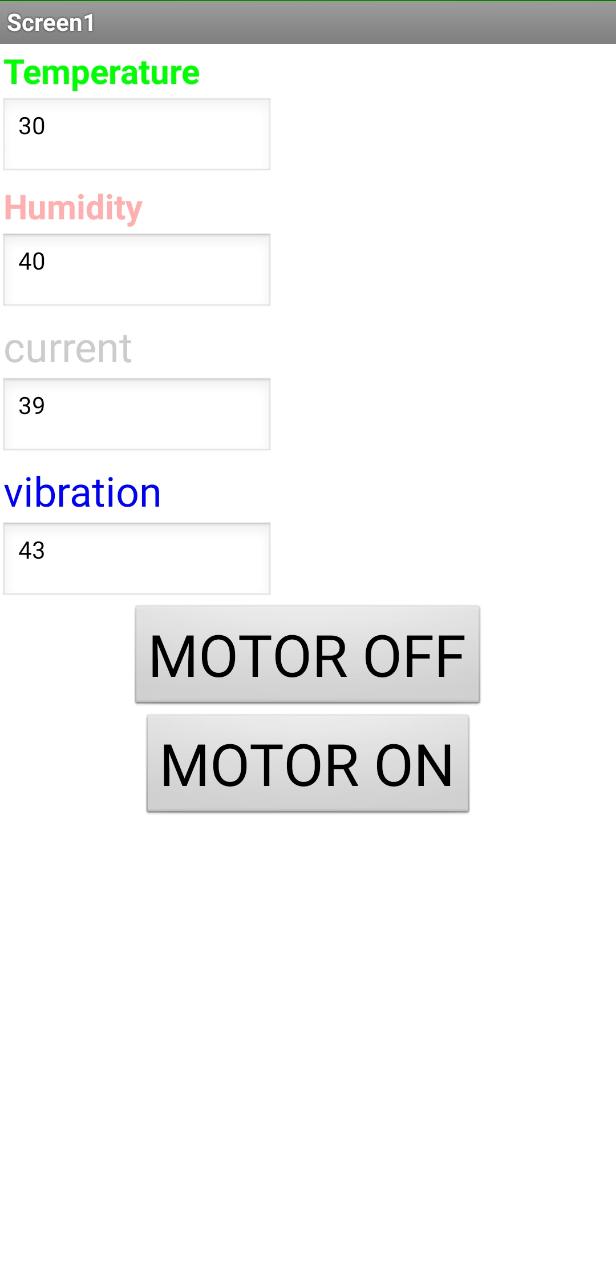


Results

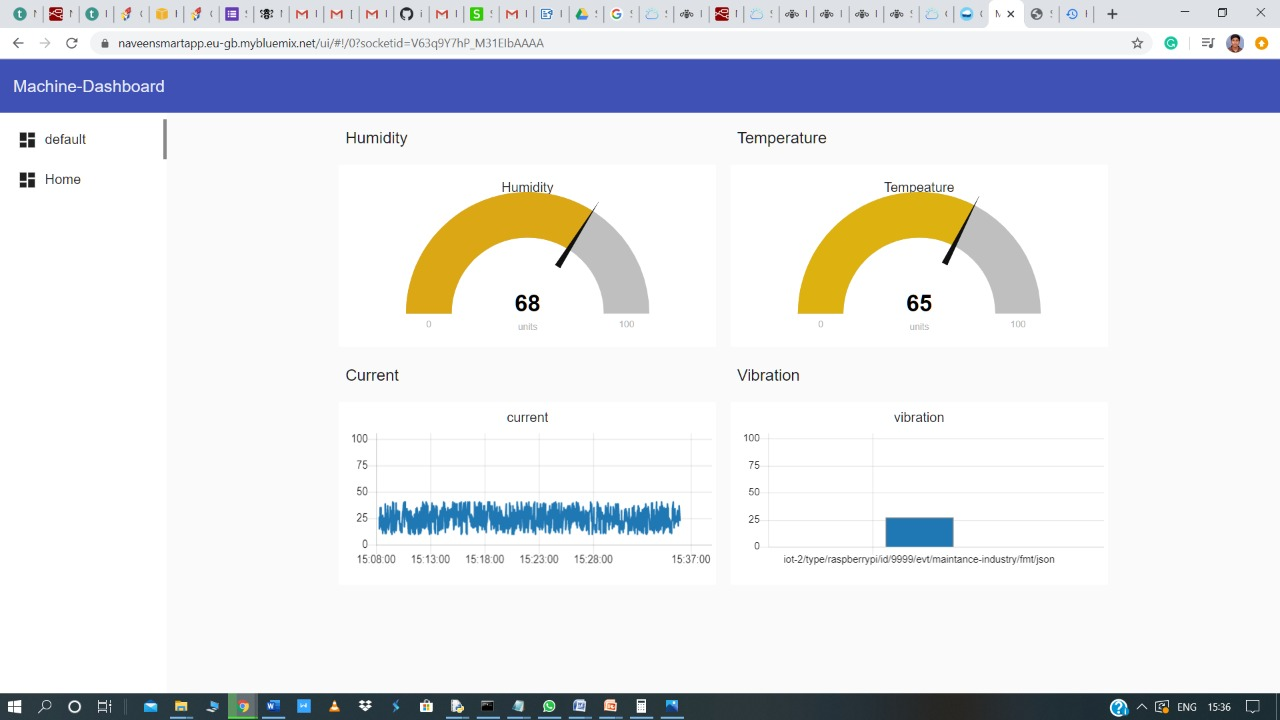
Client To App Communication Through Mit App:

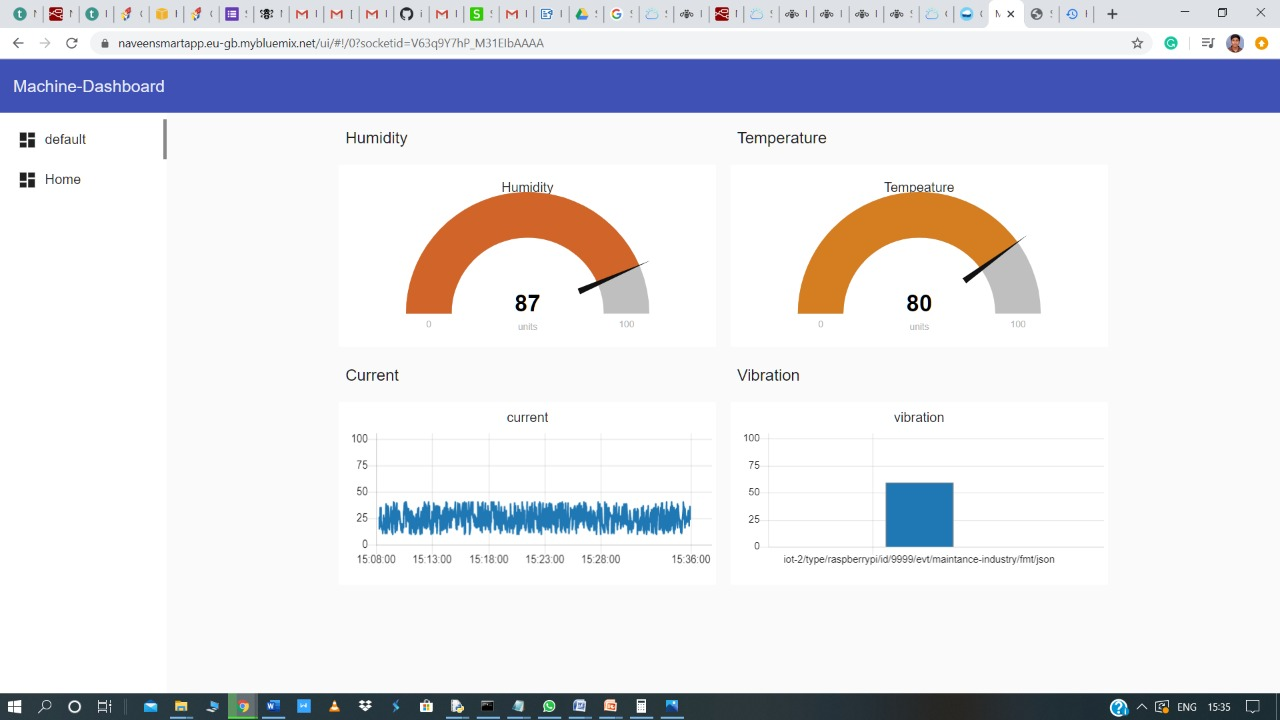


Mit App Front End:



UI Design:





**Advantages**

With enough sensors, enough data, the right software with the right algorithms, predictions on a machine can span not just in the near future, but in the far future. Condition Monitoring can hypothetically project a maintenance schedule up until the point when the machine or device is useless.

With enough historical data based on how the machine's been being used for the last six months or a year, a properly developed algorithm, or series of algorithms, can determine the maintenance schedule all the way up to the end-of-life of the machine.

The predictions could potentially tell when a machine will no longer be suitable for a manufacturing process

**Disadvantages**

The immediate reaction is often to place Condition Monitoring devices all over systems in operation, watching as many aspects of operation as one can afford. There could be thousands of sensors collecting data at any one time. This is where many companies see cost escalating.

Not every point of a machine or system needs to be monitored. Assessing what is most critical to a system, taking into account repair history and common failure points, is the best way to sidestep excessive cost.

Analyzing the cost impact of potential failures is key in deciding what to monitor

**BIBLIOGRAPGY**

>[https://developer.ibm.com/recipes/tutorials/ui-dashboard-for-iot-device-data-using-node-red](https://developer.ibm.com/recipes/tutorials/ui-dashboard-for-iot-device-data-using-node-red/)

>[http://ai2.appinventor.mit.edu](http://ai2.appinventor.mit.edu/?locale=en#)

> [https://workdrive.zohoexternal.com](https://workdrive.zohoexternal.com/)

[>](https://workdrive.zohoexternal.com/) <https://www.ibm.com/in-en/cloud>

**Appendix**

**Source CODE** (Arduino Based)

#define BLYNK\_PRINT Serial

#include <Adafruit\_BMP280.h>

#include <Adafruit\_Sensor.h>

#include <Wire.h>

#include <ESP8266\_Lib.h>

#include <BlynkSimpleShieldEsp8266.h>

#include <DHT.h>

char auth[] = "45381498098";

char ssid[] = "Naveen";

char pass[] = "Hemanttthh";

#define EspSerial Serial3

#define ESP8266\_BAUD 115200

ESP8266 wifi(&EspSerial);

#define DHTPIN 4

#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);

Adafruit\_BMP280 bmp;

BlynkTimer timer\_dht;

BlynkTimer timer\_bmp;

BlynkTimer timer\_mq2;

const int calibrationLed = 13;

const int MQ\_PIN = A0;

const int MQ\_power = 45;

int RL\_VALUE = 1;

float RO\_CLEAN\_AIR\_FACTOR = 9.83;

#define MQ\_powerPin 8

int CALIBARAION\_SAMPLE\_TIMES = 50;

int CALIBRATION\_SAMPLE\_INTERVAL = 500;

int READ\_SAMPLE\_INTERVAL = 50;

int READ\_SAMPLE\_TIMES = 5;

#define GAS\_LPG 0

#define GAS\_CO 1

#define GAS\_SMOKE 2

float LPGCurve[3] = {2.3, 0.21, -0.47};

float COCurve[3] = {2.3, 0.72, -0.34};

float SmokeCurve[3] = {2.3, 0.53, -0.44};

float Ro = 0.35;

void sendDHT()

{

float h = dht.readHumidity();

float t2 = dht.readTemperature();

if (isnan(h) || isnan(t2)) {

Serial.println("Failed to read from DHT sensor!");

return;

}

Blynk.virtualWrite(V4, h);

}

void sendBMP() {

float t = bmp.readTemperature();

float p = bmp.readPressure();

if (isnan(p) || isnan(t)) {

Serial.println("Failed to read from BMP sensor!");

return;

}

p \*= 0.00750062;

Blynk.virtualWrite(V6, t);

Blynk.virtualWrite(V5, p);

Serial.println(t);

Serial.println(p);

}

void sendMQ2() {

digitalWrite(MQ\_power,HIGH);

delay(500);

long iPPM\_LPG = 0;

long iPPM\_CO = 0;

long iPPM\_Smoke = 0;

iPPM\_LPG = MQGetGasPercentage(MQRead(MQ\_PIN) / Ro, GAS\_LPG);

iPPM\_CO = MQGetGasPercentage(MQRead(MQ\_PIN) / Ro, GAS\_CO);

iPPM\_Smoke = MQGetGasPercentage(MQRead(MQ\_PIN) / Ro, GAS\_SMOKE);

Serial.println("Concentration of gas ");

Serial.print("LPG: ");

Serial.print(iPPM\_LPG);

Serial.println(" ppm");

Serial.print("CO: ");

Serial.print(iPPM\_CO);

Serial.println(" ppm");

Serial.print("Smoke: ");

Serial.print(iPPM\_Smoke);

Serial.println(" ppm");

Blynk.virtualWrite(V8, iPPM\_CO);

Blynk.virtualWrite(V7, iPPM\_LPG);

Blynk.virtualWrite(V9, iPPM\_Smoke);

digitalWrite(MQ\_power,LOW);

delay(200);

}

void setup()

{

Serial.begin(9600);

EspSerial.begin(ESP8266\_BAUD);

delay(10);

Blynk.begin(auth, wifi, ssid, pass);

pinMode(MQ\_PIN, INPUT);

pinMode(MQ\_power, OUTPUT);

dht.begin();

bmp.begin();

timer\_dht.setInterval(15000L, sendDHT);

timer\_bmp.setInterval(9000L, sendBMP);

timer\_mq2.setInterval(12000L, sendMQ2);

}

void loop()

{

Blynk.run();

timer\_dht.run();

timer\_bmp.run();

timer\_mq2.run();

}

float MQResistanceCalculation(int raw\_adc)

{

return ( ((float)RL\_VALUE \* (1023 - raw\_adc) / raw\_adc));

}

float MQCalibration(int mq\_pin)

{

int i;

float val = 0;

for (i = 0; i < CALIBARAION\_SAMPLE\_TIMES; i++) {

val += MQResistanceCalculation(analogRead(mq\_pin));

delay(CALIBRATION\_SAMPLE\_INTERVAL);

}

val = val / CALIBARAION\_SAMPLE\_TIMES;

val = val / RO\_CLEAN\_AIR\_FACTOR;

return val;

}

float MQRead(int mq\_pin)

{

int i;

float rs = 0;

for (i = 0; i < READ\_SAMPLE\_TIMES; i++) {

rs += MQResistanceCalculation(analogRead(mq\_pin));

delay(READ\_SAMPLE\_INTERVAL);

}

rs = rs / READ\_SAMPLE\_TIMES;

return rs;

}

long MQGetGasPercentage(float rs\_ro\_ratio, int gas\_id)

{

if ( gas\_id == GAS\_LPG ) {

return MQGetPercentage(rs\_ro\_ratio, LPGCurve);

} else if ( gas\_id == GAS\_CO ) {

return MQGetPercentage(rs\_ro\_ratio, COCurve);

} else if ( gas\_id == GAS\_SMOKE ) {

return MQGetPercentage(rs\_ro\_ratio, SmokeCurve);

}

return 0;

}

long MQGetPercentage(float rs\_ro\_ratio, float \*pcurve)

{

return (pow(10, ( ((log(rs\_ro\_ratio) - pcurve[1]) / pcurve[2]) + pcurve[0])));

}

**Python code**

import time  
import sys  
import ibmiotf.application  
import ibmiotf.device  
import random  
#Provide your IBM Watson Device Credentials  
organization = "0ghxj6"  
deviceType = "raspberrypi"  
deviceId = "9999"  
authMethod = "token"  
authToken = "12345678"  
  
# Initialize GPIO  
  
def myCommandCallback(cmd):  
 print("Command received: %s" % cmd.data)  
 print(type(cmd.data))  
 i=cmd.data['command']  
 if i=='lighton':  
 print("light is on")  
 elif i=='lightoff':  
 print("light is off")  
  
try:  
 deviceOptions = {"org": organization, "type": deviceType, "id": deviceId, "auth-method": authMethod, "auth-token": authToken}  
 deviceCli = ibmiotf.device.Client(deviceOptions)#.............................................  
   
except Exception as e:  
 print("Caught exception connecting device: %s" % str(e))  
 sys.exit()  
  
# Connect and send a datapoint "hello" with value "world" into the cloud as an event of type "greeting" 10 times  
deviceCli.connect()  
  
while True:  
   
 hum=random.randint(10,90)  
 #print(hum)  
 temp =random.randint(30,80)  
 current=random.randint(10,40)  
 vibration=random.randint(10,80)  
 #Send Temperature & Humidity to IBM Watson  
 data = { 'Temperature' : temp, 'Humidity': hum,'current':current,'vibration':vibration }  
 #print (data)  
 def myOnPublishCallback():  
 print ("Published Temperature = %s C" % temp, "Humidity = %s %%" % hum,"current=%s A " % current , "vibration=%s D" % vibration ,"to IBM Watson")  
  
 success = deviceCli.publishEvent("maintance-industry", "json", data, qos=0, on\_publish=myOnPublishCallback)  
 if not success:  
 print("Not connected to IoTF")  
 time.sleep(2)  
   
 deviceCli.commandCallback = myCommandCallback